MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

ECP2216 – MICROCONTROLLER AND MICROPROCESSOR SYSTEMS

(All sections / Groups)

20 OCTOBER 2016
2.30 p.m – 4.30 p.m
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of 11 pages with 5 Questions only.
2. Attempt **ALL FIVE COMPULSORY** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided.

Question 1

(a) What are the FOUR major components in a computer system?

[2 marks]

(b) Using a suitable diagram, explain von-Neumann "Fetch-Execute" cycle.

[2 + 3 marks]

(c) List SIX criteria in selecting a micro-controller during project's initial design stage.

[3 marks]

(d) (i) Describe the difference(s) between the Multicore Processor Technology and Multi-processor System.

[4 marks]

(ii) Briefly describe the function of prefetch unit, decode unit, execution unit, segmentation unit, paging unit and bus unit in 80386DX Intel 32-bit microprocessor.

[6 marks]

Continued ...

Question 2

(a) What is the function for the following pins in 8051?

i) RST [1 Mark]
ii) EA [1 Mark]
iii) ALE [1 Mark]
iv) PSEN [1 Mark]

(b) Design an 8051-based system that supports 16K bytes of external data memory. Assume that the 8051 internal ROM is used, and only 4K bytes x 8-bit memory ICs are available) [10 Marks]

(c) Program Status Word (PSW) is an important Special Function Registers (SFR) that reflects the current state of Central Processing Unit (CPU).
Assuming the initial contents of the registers are,

[A] = 00H
[PSW] = 45H
[R0] = 33H
[R1] = 88H

Evaluate the content of PSW and Accumulator (ACC) after the execution of the following instructions in sequence.

i) MOV A, #0A5 [2 Mark]
ii) ORL A, R0 [2 Mark]
iii) ADD A, R1 [2 Mark]

Continued ...

Question 3

(a) Evaluate the addressing mode used for the source and destination operand in the following instructions.

i) MOV R0, #0A5H [2 Marks]

ii) ORL A, @R0 [2 Marks]

iii) MOV R1, @A+DPTR [2 Marks]

(b) Using MCS-51 instructions sets, write a complete program for 8051 that performs,

i) Initialize P1.0 to be a digital input.

ii) Read P1.0 at every 1ms interval.

iii) If P1.0 logic is HIGH, call subroutine "ALERT", else clear P2.0 and P2.1.

iv) In subroutine "ALERT", set P2.0 and P2.1.

[14 Marks]

Assume 12 MHz crystal clock is used.

Continued ...

Question 4

(a) Define interrupt service routine (ISR) and discuss the difference between small ISR and large ISR.

[5 marks]

(b) Write a MCS-51 assembly language subroutine to repetitively transmit an ASCII character 'P' (50H) followed by an ASCII character 'A' (41H) via serial port. *Assume 11.0592MHz crystal frequency, SMOD = 1, 8-bit data, 1 stop bit, no parity and operation at 4800 baud rate generated by Timer 1.*

[6 marks]

(c) Write a MCS-51 assembly language program to perform the following tasks using interrupts.

An 8051 microcontroller system with external interrupt INT1 pin and INT0 pin connected to switch 1 and switch 0 respectively. The switches are normally HIGH.

- ***When switch 1 goes low, timer 1 interrupt generates a pulse width of 0.75ms from P2.4.***
- ***When switch 0 goes low, timer 0 interrupt generates a pulse width of 0.5ms from P2.5.***

[9 marks]

Continued ...

Question 5

(a)

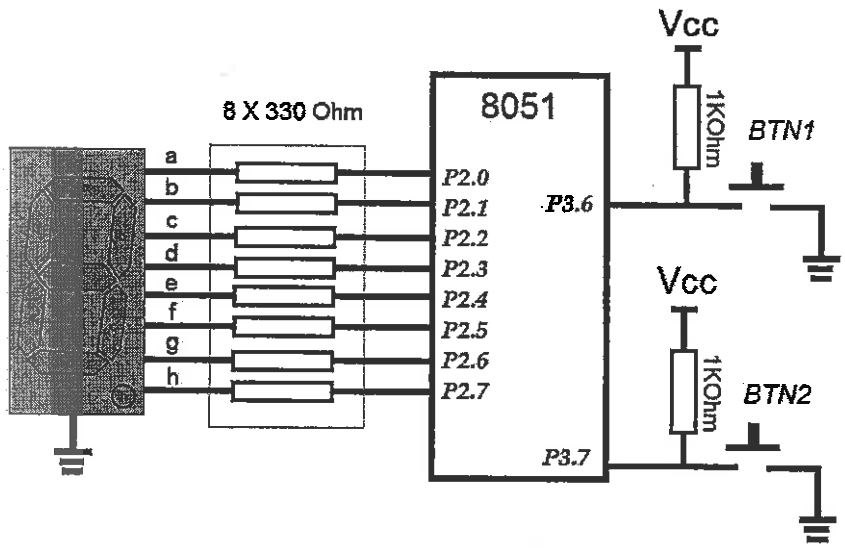


Figure Q5(a)

Figure Q5(a) shows an 8051 microcontroller interfaces to two press buttons BTN1 and BTN2 on P3.6 and P3.7 respectively, and a seven-segment LED display device on Port 2.

(i) Determine whether the seven-segment LED display is common anode-type or common cathode-type.

[1 mark]

(ii) Given the bit patterns for each character to decode the seven-segment LED display as in Table Q5(a),

Table Q5(a)

	P2.7	*P2.6*	*P2.5*	*P2.4*	*P2.3*	*P2.2*	*P2.1*	*P2.0*
Character	h	g	f	e	d	c	b	a
A	0	1	1	1	0	1	1	1
E	0	1	1	1	1	0	0	1
S	0	1	1	0	1	1	0	1
P	0	1	1	1	0	0	1	1
Y	0	1	1	0	1	1	1	0

Continued ...

Question 5

Write a MCS-51 assembly language program to perform the following tasks:

- ***When the program starts, wait for the button press on BTN1.***
- ***Once the BTN1 is pressed, seven-segment LED display will repeatedly display the characters in sequence starting from E, A, S, Y, P, A, S, and S.*** *(Time duration for each character to be displayed is 1 second.)*
- ***The display will be stopped only if the button press on BTN2 occurs.***

Assume 1 second delay subroutine ONEDELAY is available.

[7 marks]

(b) An 8051 microcontroller based ***Automated Premix White Coffee Mix-and-Dispense Machine*** is shown in Figure Q5(b).

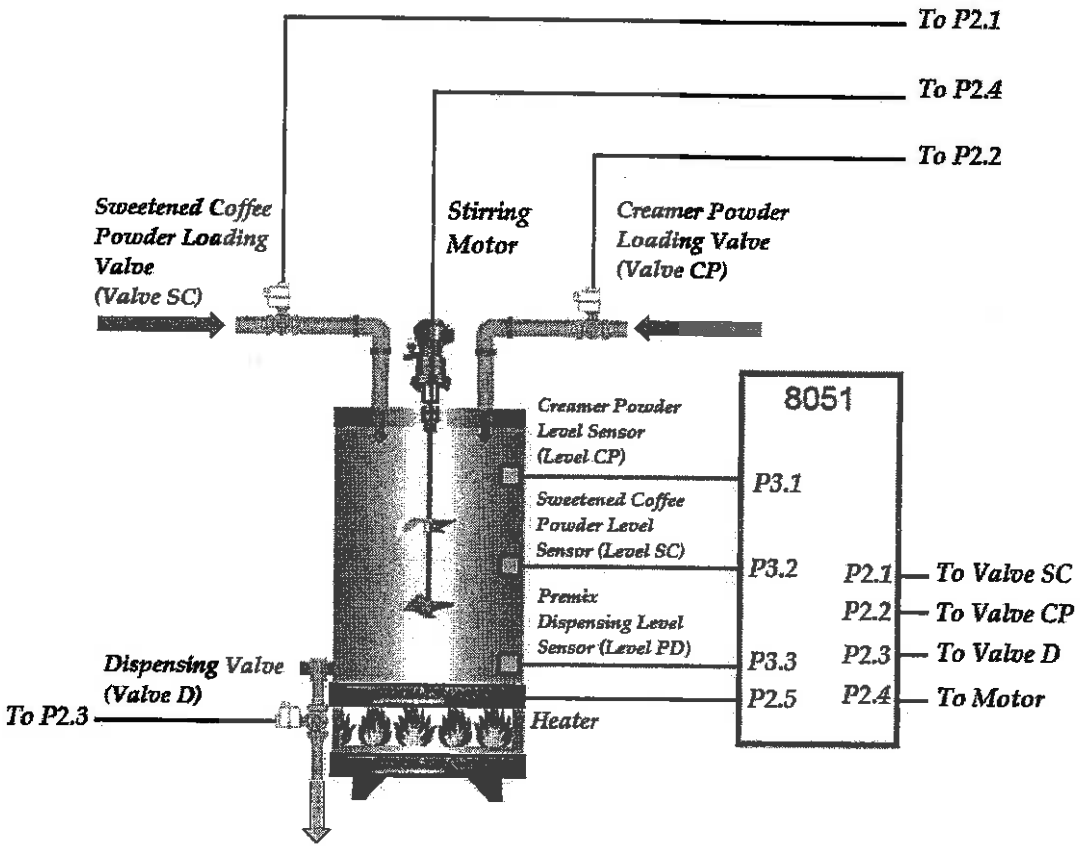


Figure Q5(b)

Continued ...

The machine performs the following process sequences:

1. The tank is first loaded with sweetened coffee powder through a solenoid Valve SC.
2. When the sweetened coffee powder reaches Level SC, Valve SC is closed and the tank is now loaded with creamer powder through Valve CP.
3. When the mixture in the tank reaches Level CP, Valve CP is closed.
4. The stirring motor starts the stirring process that last for approximately 1 minute.
5. After that, heater is turned for heating process last for approximately 2 minutes.
6. Then, the stirring motor starts the stirring process again that last for approximately 1 minute.
7. After that, the dispensing Valve D opens to drain the mixture.
8. When the mixture reaches Level PD, Valve D is closed.
9. The whole process is repeated from Step 1 again.

- The tank has three level sensors that send signals to input lines ***P3.1*** to ***P3.3***. A logical **LOW** from the sensor indicates that the level has been reached.

- The output lines ***P2.1***, ***P2.2***, and ***P2.3*** provide signals to the solenoid valves. A logical **HIGH** from the lines will open the corresponding valve.

- The output lines ***P2.4*** and ***P2.5*** provide signals to the stirring motor and heater respectively which are both activated by a logical **HIGH**.

Write a MCS-51 assembly language program to carry out the process. Assume 12MHz crystal frequency is used

[12 marks]

End of Page

APPENDIX

Special Function Register Formats

Interrupt Enable (IE)

Bit Addr.	AFH	-	-	ACH	ABH	AAH	A9H	A8H
Name	EA	-	-	ES	ET1	EX1	ET0	EX0

BIT	SYMBOL	FUNCTION (Enable=1, Disable=0)
IE.7	EA	Global enable/disable. EA = 1, each individual source is enabled/disabled by setting/clearing its enable bit. EA = 0, disable all interrupts.
IE.6	-	Undefined
IE.5	-	Not implemented in 8051. ET2 for 8052.
IE.4	ES	Serial port interrupt enable bit.
IE.3	ET1	Timer 1 interrupt enable bit.
IE.2	EX1	External interrupt enable bit.
IE.1	ET0	Timer 0 interrupt enable bit.
IE.0	EX0	External interrupt enable bit.

Interrupt Priority (IP)

Bit Addr.	-	-	-	BCH	BBH	BAH	B9H	B8H
Name	-	-	-	PS	PT1	PX1	PT0	PX0

BIT	SYMBOL	FUNCTION (Enable=1, Disable=0)
IP.7	-	Undefined
IP.6	-	Undefined
IP.5	-	Not implemented in 8051. PT2 for 8052.
IP.4	PS	Serial port interrupt priority bit.
IP.3	PT1	Timer 1 interrupt priority bit.
IP.2	PX1	External interrupt priority bit.
IP.1	PT0	Timer 0 interrupt priority bit.
IP.0	PX0	External interrupt priority bit.

Interrupt Vectors

Interrupt Source	Flag	Vector Address
System Reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial Port	RI & TI	0023H
Timer 2 (8052)	TF2 or EXF2	002BH

Program Status Word (PSW)

Bit Addr.	D7H	D6H	D5H	D4H	D3H	D2H	-	D0H
Name	CY	AC	F0	RS1	RS0	0V	-	P

Serial Control (SCON)

Bit Addr.	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H
Name	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

BIT	SYMBOL	FUNCTION
SCON.7	SM0	Serial port mode bit 0 (see Table A.1).
SCON.6	SM1	Serial port mode bit 1 (see Table A.1).
SCON.5	SM2	Serial port mode bit 2; enables multiprocessor communications in modes 2 and 3; RI will not be activated if received 9th bit is 0. In mode 1, if SM2 = 1, then RI will be activated only if a valid stop bit was received. In mode 0, SM2 should be 0.
SCON.4	REN	Receiver enable; must be set to receive characters.
SCON.3	TB8	Transmit bit 8; 9th bit transmitted in modes 2 and 3; set/cleared by software.
SCON.2	RB8	Receive bit 8; 9th bit received.
SCON.1	TI	Transmit interrupt flag; set at end of character transmission; cleared by software.
SCON.0	RI	Receive interrupt flag; set at end of character reception; cleared by software.

Table A.1 The 8051 Serial Port Mode Selection

SM0	SM1	Mode	Description	Baud Rate
0	0	0	Shift register	Fixed
0	1	1	8-bit UART	Variable
1	0	2	9-bit UART	Fixed
1	1	3	9-bit UART	Variable

Timer Control (TCON)

Bit Addr.	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
Name	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

BIT	SYMBOL	FUNCTION
TCON.7	TF1	Timer-1 overflow flag. Set by hardware on overflow. Cleared by hardware when processor vectors to interrupt routine. Must be cleared by software when not involve interrupt
TCON.6	TR1	Timer-1 run control bit. Set/cleared by software to turn timer/counter on/off.
TCON.5	TF0	Timer-0 overflow flag. Do the same function as TF1 but for Timer-0.
TCON.4	TR0	Timer-0 run control bit. Do the same function as TR1 but for Timer-0.
TCON.3	IE1	External interrupt-1 edge flag. Set by hardware when interrupt-1 falling edge is detected. Cleared by hardware when interrupt is processed.
TCON.2	IT1	Interrupt-1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
TCON.1	IE0	External interrupt-1 edge flag. Do the same function as IE1 but for external interrupt-0.
TCON.0	IT0	Interrupt-0 Type control bit. Do the same function as IT1 but for external interrupt-0.

Timer Mode (TMOD)

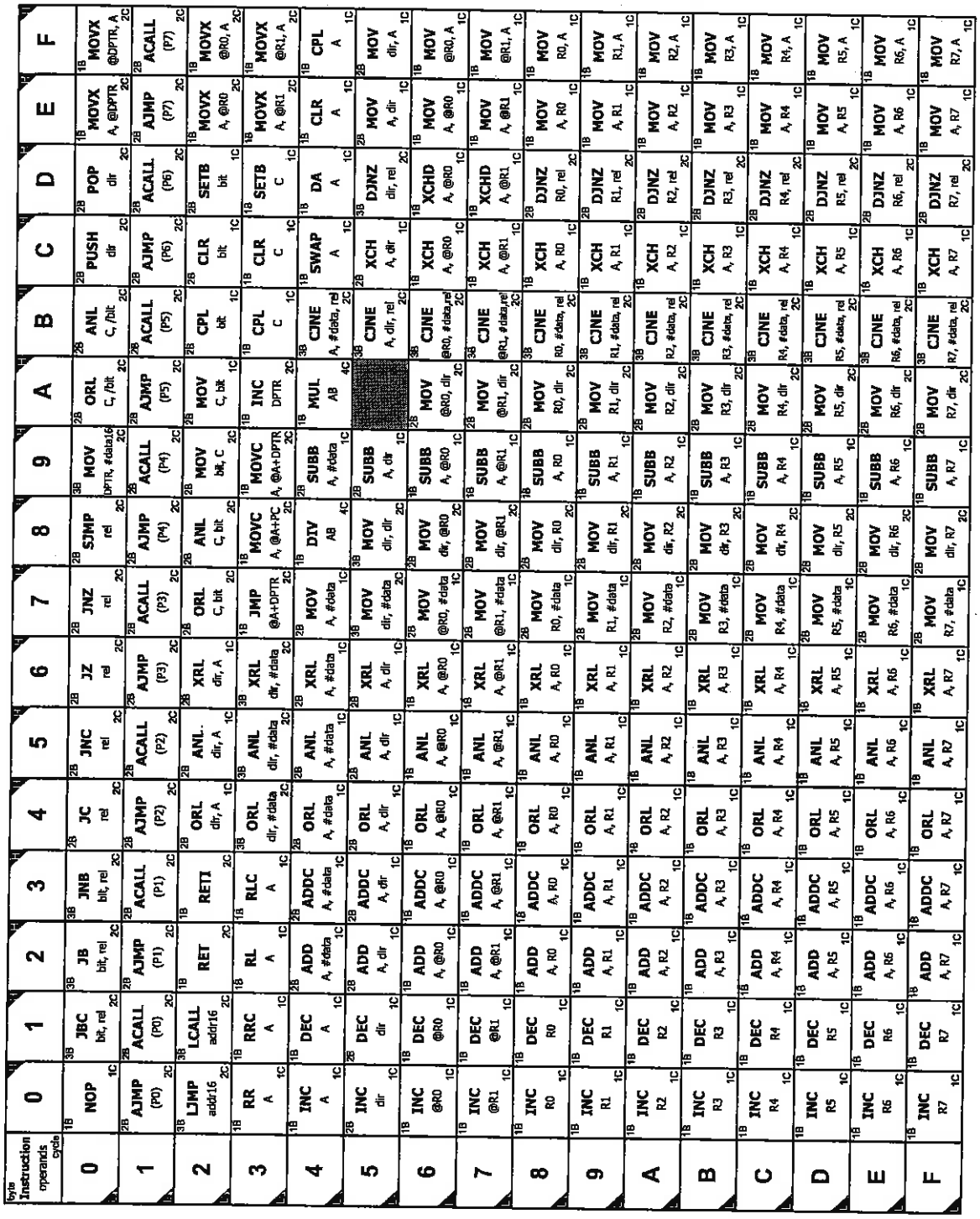
Bit	7	6	5	4	3	2	1	0
Name	GATE	C/T	M1	M0	GATE	C/T	M1	M0

BIT	SYMBOL	FUNCTION
TMOD.7	GATE1	When this bit is set the timer will only run when INT1 (P3.3) is high (hardware control). When this bit is cleared the timer will run regardless of the state of INT1 (software control).
TMOD.6	C/T1	Timer / Counter select bit. C / $\overline{T}$ = 0 → Timer operation. C / $\overline{T}$ = 1 → Counter operation.
TMOD.5	M1	Mode selection bits (see Table A.2). [for timer 1]
TMOD.4	M0	Mode selection bits (see Table A.2). [for timer 1]
TMOD.3	GATE0	Exactly the same function as GATE1 but for Timer0
TMOD.2	C/T0	Exactly the same function as C/T1 but for Timer0
TMOD.1	M1	Mode selection bits (see Table A.2). [for timer 0]
TMOD.0	M0	Mode selection bits (see Table A.2). [for timer 0]

Table A.2 Timer Mode Selection

M1	M0	Timer Mode	Description of Mode
0	0	0	13-bit Timer
0	1	1	16-bit Timer
1	0	2	8-bit auto-reload
1	1	3	Split timer mode

MCS-51 Opcode Map


byte
Instruction
operands
cycle
0 NOP 1C | JBC bit, rel 3B 2C | JB bit, rel 3B 2C | JNB bit, rel 3B 2C | JC rel 2B 2C | JNC rel 2B 2C | JZ rel 2B 2C | JNZ rel 2B 2C | SJMP rel 2B 2C | MOV DPTR, #data16 3B 2C | ORL C, /bit 2B 2C | ANL C, /bit 2B 2C | PUSH dir 2B 2C | POP dir 2B 2C | MOVX A, @DPTR 1B 2C | MOVX @DPTR, A 1B 2C
1 AJMP (P0) 2B 2C | ACALL (P0) 2B 2C | AJMP (P1) 2B 2C | ACALL (P1) 2B 2C | AJMP (P2) 2B 2C | ACALL (P2) 2B 2C | AJMP (P3) 2B 2C | ACALL (P3) 2B 2C | AJMP (P4) 2B 2C | ACALL (P4) 2B 2C | AJMP (P5) 2B 2C | ACALL (P5) 2B 2C | AJMP (P6) 2B 2C | ACALL (P6) 2B 2C | AJMP (P7) 2B 2C | ACALL (P7) 2B 2C
2 LJMP addr16 3B 2C | LCALL addr16 3B 2C | RET 1B 2C | RETI 1B 2C | ORL dir, A 2B 1C | ANL dir, A 2B 1C | XRL dir, A 2B 1C | ORL C, bit 2B 2C | ANL C, bit 2B 2C | MOV bit, C 2B 2C | MOV C, bit 2B 1C | CPL bit 2B 1C | CLR bit 2B 1C | SETB bit 2B 1C | MOVX A, @R0 1B 2C | MOVX @R0, A 1B 2C
3 RR A 1B 1C | RRC A 1B 1C | RL A 1B 1C | RLC A 1B 1C | ORL dir, #data 3B 2C | ANL dir, #data 3B 2C | XRL dir, #data 3B 2C | JMP @A+DPTR 1B 2C | MOV A, #data 2B 1C | MOVC A, @A+PC 1B 2C | INC DPTR 1B 2C | CPL C 1B 1C | CLR C 1B 1C | SETB C 1B 1C | MOVX A, @R1 1B 2C | MOVX @R1, A 1B 2C
4 INC A 1B 1C | DEC A 1B 1C | ADD A, #data 2B 1C | ADDC A, #data 2B 1C | ORL A, #data 2B 1C | ANL A, #data 2B 1C | XRL A, #data 2B 1C | MOV dir, #data 3B 2C | DIV AB 1B 4C | SUBB A, #data 2B 1C | MUL AB 1B 4C | CJNE A, #data, rel 3B 2C | SWAP A 1B 1C | DA A 1B 1C | CLR A 1B 1C | CPL A 1B 1C
5 INC dir 2B 1C | DEC dir 2B 1C | ADD A, dir 2B 1C | ADDC A, dir 2B 1C | ORL A, dir 2B 1C | ANL A, dir 2B 1C | XRL A, dir 2B 1C | MOV dir, dir 3B 2C | MOV dir, @R0 2B 2C | SUBB A, dir 2B 1C | CJNE A, dir, rel 3B 2C | XCH A, dir 2B 1C | DJNZ dir, rel 3B 2C | MOV A, dir 2B 1C | MOV dir, A 2B 1C
6 INC @R0 1B 1C | DEC @R0 1B 1C | ADD A, @R0 1B 1C | ADDC A, @R0 1B 1C | ORL A, @R0 1B 1C | ANL A, @R0 1B 1C | XRL A, @R0 1B 1C | MOV @R0, #data 2B 1C | MOV dir, @R0 2B 2C | SUBB A, @R0 1B 1C | MOV @R0, dir 2B 2C | CJNE @R0, #data,rel 3B 2C | XCH A, @R0 1B 1C | XCHD A, @R0 1B 1C | MOV A, @R0 1B 1C | MOV @R0, A 1B 1C
7 INC @R1 1B 1C | DEC @R1 1B 1C | ADD A, @R1 1B 1C | ADDC A, @R1 1B 1C | ORL A, @R1 1B 1C | ANL A, @R1 1B 1C | XRL A, @R1 1B 1C | MOV @R1, #data 2B 1C | MOV dir, @R1 2B 2C | SUBB A, @R1 1B 1C | MOV @R1, dir 2B 2C | CJNE @R1, #data,rel 3B 2C | XCH A, @R1 1B 1C | XCHD A, @R1 1B 1C | MOV A, @R1 1B 1C | MOV @R1, A 1B 1C
8 INC R0 1B 1C | DEC R0 1B 1C | ADD A, R0 1B 1C | ADDC A, R0 1B 1C | ORL A, R0 1B 1C | ANL A, R0 1B 1C | XRL A, R0 1B 1C | MOV R0, #data 2B 1C | MOV dir, R0 2B 2C | SUBB A, R0 1B 1C | MOV R0, dir 2B 2C | CJNE R0, #data, rel 3B 2C | XCH A, R0 1B 1C | DJNZ R0, rel 2B 2C | MOV A, R0 1B 1C | MOV R0, A 1B 1C
9 INC R1 1B 1C | DEC R1 1B 1C | ADD A, R1 1B 1C | ADDC A, R1 1B 1C | ORL A, R1 1B 1C | ANL A, R1 1B 1C | XRL A, R1 1B 1C | MOV R1, #data 2B 1C | MOV dir, R1 2B 2C | SUBB A, R1 1B 1C | MOV R1, dir 2B 2C | CJNE R1, #data, rel 3B 2C | XCH A, R1 1B 1C | DJNZ R1, rel 2B 2C | MOV A, R1 1B 1C | MOV R1, A 1B 1C
A INC R2 1B 1C | DEC R2 1B 1C | ADD A, R2 1B 1C | ADDC A, R2 1B 1C | ORL A, R2 1B 1C | ANL A, R2 1B 1C | XRL A, R2 1B 1C | MOV R2, #data 2B 1C | MOV dir, R2 2B 2C | SUBB A, R2 1B 1C | MOV R2, dir 2B 2C | CJNE R2, #data, rel 3B 2C | XCH A, R2 1B 1C | DJNZ R2, rel 2B 2C | MOV A, R2 1B 1C | MOV R2, A 1B 1C
B INC R3 1B 1C | DEC R3 1B 1C | ADD A, R3 1B 1C | ADDC A, R3 1B 1C | ORL A, R3 1B 1C | ANL A, R3 1B 1C | XRL A, R3 1B 1C | MOV R3, #data 2B 1C | MOV dir, R3 2B 2C | SUBB A, R3 1B 1C | MOV R3, dir 2B 2C | CJNE R3, #data, rel 3B 2C | XCH A, R3 1B 1C | DJNZ R3, rel 2B 2C | MOV A, R3 1B 1C | MOV R3, A 1B 1C
C INC R4 1B 1C | DEC R4 1B 1C | ADD A, R4 1B 1C | ADDC A, R4 1B 1C | ORL A, R4 1B 1C | ANL A, R4 1B 1C | XRL A, R4 1B 1C | MOV R4, #data 2B 1C | MOV dir, R4 2B 2C | SUBB A, R4 1B 1C | MOV R4, dir 2B 2C | CJNE R4, #data, rel 3B 2C | XCH A, R4 1B 1C | DJNZ R4, rel 2B 2C | MOV A, R4 1B 1C | MOV R4, A 1B 1C
D INC R5 1B 1C | DEC R5 1B 1C | ADD A, R5 1B 1C | ADDC A, R5 1B 1C | ORL A, R5 1B 1C | ANL A, R5 1B 1C | XRL A, R5 1B 1C | MOV R5, #data 2B 1C | MOV dir, R5 2B 2C | SUBB A, R5 1B 1C | MOV R5, dir 2B 2C | CJNE R5, #data, rel 3B 2C | XCH A, R5 1B 1C | DJNZ R5, rel 2B 2C | MOV A, R5 1B 1C | MOV R5, A 1B 1C
E INC R6 1B 1C | DEC R6 1B 1C | ADD A, R6 1B 1C | ADDC A, R6 1B 1C | ORL A, R6 1B 1C | ANL A, R6 1B 1C | XRL A, R6 1B 1C | MOV R6, #data 2B 1C | MOV dir, R6 2B 2C | SUBB A, R6 1B 1C | MOV R6, dir 2B 2C | CJNE R6, #data, rel 3B 2C | XCH A, R6 1B 1C | DJNZ R6, rel 2B 2C | MOV A, R6 1B 1C | MOV R6, A 1B 1C
F INC R7 1B 1C | DEC R7 1B 1C | ADD A, R7 1B 1C | ADDC A, R7 1B 1C | ORL A, R7 1B 1C | ANL A, R7 1B 1C | XRL A, R7 1B 1C | MOV R7, #data 2B 1C | MOV dir, R7 2B 2C | SUBB A, R7 1B 1C | MOV R7, dir 2B 2C | CJNE R7, #data, rel 3B 2C | XCH A, R7 1B 1C | DJNZ R7, rel 2B 2C | MOV A, R7 1B 1C | MOV R7, A 1B 1C